ID36- DATA ACQUISITION SYSTEM DEVELOPMENT FOR EGIM ON EMSODEV EU PROJECT

ÓSCAR GARCÍA[210], JUANJO DAÑOBEITIA[206], JORDI SORRIBAS[209], RAQUEL CASAS[211], JAUME PIERA[110], RAFAEL BARTOLOME[102], RAÚL BARDAJI[111], JOAQUÍN DEL RIO[182], JAVIER CADENA[180], DANIEL MIHAL[181], IKRAM BGHIEL[179], ENOC MARTÍNEZ[178], MARC NOGUERAS[222]

Abstract – The EMSODEV1 (European Multidisciplinary Seafloor and water--column Observatory DEVelopment) is a UE project whose general objective is to set up the full implementation and operation of the EMSO distributed Research Infrastructure (RI), through the development, testing and deployment of an EMSO Generic Instrument Module (EGIM). The EGIM module will measure various ocean parameters in a long-term consistent, accurate and comparable manner. These measurements are critical to respond accurately to the social and scientific challenges such as climate change, changes in marine ecosystems, and marine hazards. Here we present the current status of the EGIM data acquisition system development

Keywords – EMSO, data acquisition, EMSODEV, EGIM, OGC, SOS, SES, SWE, 52 North , SensorML, Zabbix.

I. INTRODUCTION

The general objective of the EMSODEV project is to implement a Generic Sensor Module (EGIM) within the EMSO ERIC distributed marine research infrastructure. Our contribution in the implementation of the EGIM data acquisition system module (WP4 of the EMSODEV project) focusses on developing standard-compliant generic software for Sensor Web Enablement. The main objective of this development is to allow the registration and data acquisition of a new EGIM system and all sensors connected to it, to a centralised SOS server and to a laboratory monitor system (LabMonitor) for recording events and alarms.

The primary aim is to validate the EGIM prototype system in shallow-water conditions. The acquisition layer is located in the structure of the system between the data source (EGIM) and the data management. Thus, two main interfaces are implemented. On one side, they should guarantee that the data has been recorded from the EGIM hardware and for the other side, it must implement a standardize OGC-SWE compliant SOS2 Gateway to allow polling data from data-management layer.

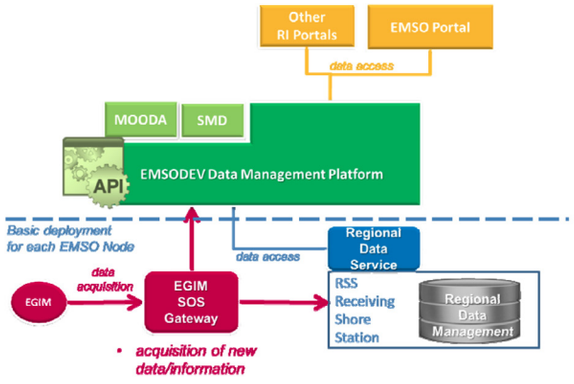

Figure 1 EGIM diagram showing SOS Gateway relations

II. SYSTEM OVERVIEW

In order to get all the hardware required by each components of this system we use a virtualization service, that virtualizes the hardware for each separate roll on the whole system generating three virtual machines ('Mussel', 'SeaShell' and 'Donax').

We have made tree virtual machines; one for acquisition, another for SOS server and a third one for laboratory monitoring. Each one has been configured with the necessary resources (database container, web server, VPN client ...) provided with the necessary interfaces to communicate with the others hosts.

III. SOS GATEWAY INTERFACE

To accomplish the SOS Gateway requirement, and once we have analysed the

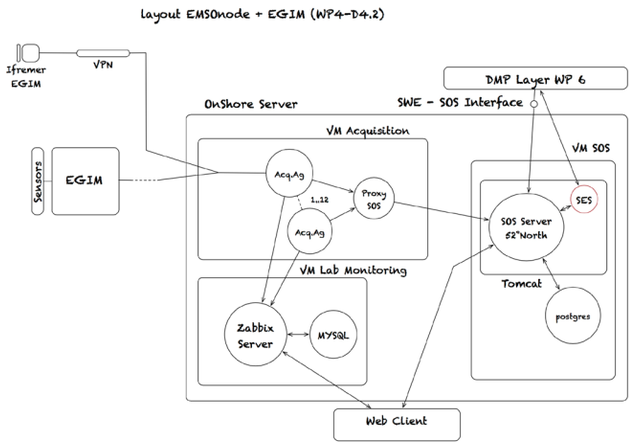

Figure 2 General Server and Services Layout

actual state of the art on SOS implementations, we have decided to use the 52 North SOS 2.0 implementation. This SOS implementation aggregates readings from live, in-situ and remote sensors. The service provides an interface to make sensors and sensor data archives both accessible through an interoperable web-based interface, using SensorML and Observation and Measurements (O&M). The main SOS 2.0 interfaces offered with this implementation are:

Core Extension
- GetCapabilities, for requesting a self-description of the service
- GetObservation, for requesting the pure sensor data encoded in Observations & Measurements 2.0 (O&M)
- DescribeSensor for requesting information about a certain sensor, encoded in a Sensor Model Language 1.0.1 (SensorML) instance document.

Enhanced Extension
- GetFeatureOfInterest, for requesting the GML 3.2.1 encoded representation of the feature that is the target of the observation. • GetObservaitonById, for requesting the pure sensor data for a specific observation identifier

Transactional Extension
- InsertSensor, for publishing new sensors
- UpdateSensorDescription, for updating the description of a sensor
- DeleteSensor, for deleting a sensor
- InsertObservation, for publishing observations for registered sensors

Result Handling Extension
- InsertResultTemplate, for inserting a result template into a SOS server that describes the structure of the values of a InsertResult of GetResult request.
- InsertResult, for uploading raw values accordingly to the structure and encoding defined in the InsertResultTemplate request
- GetResultTemplate, for getting the result structure and encoding for specific parameter constellations
- GetResult, for getting the raw data for specific parameter constellations

For this deployment we use the 'Mussel virtual machine'. Therefore we deployed the 52 North web applications over a Tomcat web server container version 7, configured with a Postgres database container. Some other small configurations have been implemented for conditioning the application in our domain.

For attending client requests, we have installed the JavaScript web client application from 52 North, which allows, in a lightweight manner, real time and historical data visualization using the SOS gateway. This web application has been opened to be accessed from outside the local network.

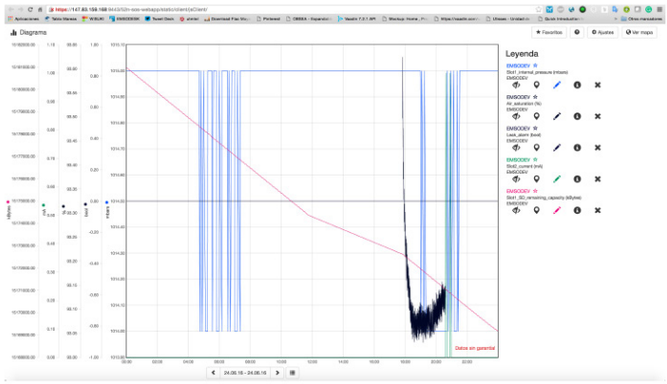

Figure 3 Visualization of EGIM status data on 52 North JavaScript web client

IV. ACQUISITION SERVICES

The acquisition environment has been deployed on the 'Seashell' virtual machine. On this server all the elements are configured to accomplish the acquisition requirements. This requirements includes the processes to acquire the data from the connected sensors to EGIM, we call this the acquisition agent, and to send the observations to SOS Gateway and to Lab Monitor systems.

To acquire data from EGIM system, we need to distinguish two kinds of reading procedures. First, there is a reading procedure for the external sensors connected to EGIM system at channels 1 to 12. This is done based on TCP Sockets connections. Second, there is a reading procedure of the internal sensors of EGIM, which is done based on readings of UPD packets to a specific port. Once the agent read this two kind of data, we use a 'proxy SOS' implemented tool, which is doing automatically all the write operation between the acquisition agent and the SOS Server. Hence, this tool is registering any new sensors connected to EGIM and send the InsertResult queries for each new data acquired from EGIM.

Moreover, the acquisition agent is generating also JSON requests to Zabbix server, in order to add these values to the database of the Lab Monitor.

```
oscar@seashell:~$ sudo tcpdump -i ens192 udp port 54320 -vv -X
[sudo] password for oscar:
tcpdump: listening on ens192, link-type EN10MB (Ethernet), capture size 262144 bytes
13:08:26.275723 IP (tos 0x0, ttl 236, id 89, offset 0, flags [none], proto UDP (17), length 84)
    134.246.149.26.49242 > seashell.54320: [udp sum ok] UDP, length 56
        0x0000:  4500 0054 0059 0000 ec11 7f2f 86f6 951a  E..T.Y...../....
        0x0010:  9353 9fac c05a d430 0040 be86 0200 3f54  .S...Z.0.@....?T
        0x0020:  2f00 0000 1006 1a0d 0800 db65 b700 0100  /..........e....
        0x0030:  0003 0800 002f 01f9 0320 25e7 0009 0000  ...../...%.....
        0x0040:  2701 f903 e036 e700 0fa2 0031 01f9 0330  '....6.....1...0
        0x0050:  f805 0003                                ....
13:08:55.959535 IP (tos 0x0, ttl 236, id 91, offset 0, flags [none], proto UDP (17), length 84)
    134.246.149.26.49244 > seashell.54320: [udp sum ok] UDP, length 56
        0x0000:  4500 0054 005b 0000 ec11 7f2d 86f6 951a  E..T.[.....-....
        0x0010:  9353 9fac c05c d430 0040 b965 0200 3f54  .S...\.0.@.e..?T
        0x0020:  2f00 0000 1006 1a0d 081e de65 b900 0100  /..........e....
        0x0030:  0003 0800 002f 01f9 0320 25e7 0009 0000  ...../...%.....
        0x0040:  2701 f903 e036 e700 0fa3 0031 01f9 0330  '....6.....1...0
        0x0050:  f805 0003                                ....
```

```
oscar@seashell:~$ sudo telnet 134.246.149.26 8022
Trying 134.246.149.26...
Connected to 134.246.149.26.
Escape character is '^]'.
%4330   2132    245.501 93.734  24.176  29.031  29.031  37.834  8.803   904.2   932.8   47.5
4330    2132    245.517 93.738  24.175  29.031  29.031  37.829  8.798   904.1   934.2   47.6
4330    2132    245.672 93.796  24.174  29.024  29.024  37.813  8.789   904.2   931.2   47.6
4330    2132    245.627 93.778  24.174  29.026  29.026  37.810  8.784   904.1   929.5   47.6
4330    2132    245.671 93.793  24.173  29.025  29.025  37.808  8.783   904.0   928.3   47.6
4330    2132    245.636 93.779  24.173  29.026  29.026  37.807  8.780   903.9   927.5   47.7
4330    2132    245.682 93.796  24.172  29.025  29.025  37.804  8.779   903.9   926.7   47.7
```

Figure 4 UDP and TCP data from EGIM

V. MONITOR LAB

To perform the benchmark test as well as for production processes, it is necessary to visualize some real data and historical trend data from sensors, with the objective to control some critical information by arranging some triggers. In the same manner, we need to monitor the correct behaviour of the whole system (EGIM Status data). For accomplishing this requirement, we have built a Lab-Monitor system, using the Zabbix4 open source application.

Zabbix is designed for monitoring availability and performance of IT infrastructure components. It works like a centralised monitoring system using active or passive agents for requesting or receiving from hosts. The system can use many protocols. In our scenario we use Zabbix agents to retrieve information about each virtual machine, and the observations from EGIM system retrieved by the acquisition agents..

For this purpose, we have created the 'Donax' virtual machine, which uses MySQL database as a data container and an apache web container for attending web client requests. In each server, it has been installed a binary Zabbix agent that is reporting all the information about the host to the Zabbix server. This functionality has been added also inside the acquisition software, to send all the data acquired from the EGIM system. The acquisition agent, gets the data received from EGIM sensors, and send it to the Zabbix server using a formatted JSON request. Then, the server informs to the client if the data has been created successfully or it sends a report in case of any problem arise. Once the data has been received on LabMonitor, the data is written to the database, and can be visualised on the web application. If a trigger has been configured for this data, the system will check the rule configuration and informs of any status change.

We can also check the state of the EGIM equipment monitoring all the status values that come from EGIM status packed. We have configured alarms for critical data as the internal temperature, internal humidity, power consumption and water leak. In case of any reporting alarm occurs, we arranged a email account to receive a message every time the trigger switch on or off in the system, informing the sensor data involved and some more detailed information.

Finally, we set up a public access for remote monitoring purposes, which only require a web client for real time system data access. Moreover, it is also feasible to request historical data, which could be really useful for analysing the events processes and crossing data.

VI. CONCLUSIONS

At the time of writing this document, we are in development phase and compiling all the necessary components for the final production environment (estimated to October 2016). For this reason, it may happen the data will have no much meaning or eventually some gaps on historical trends could appear. In the following months new development may introduce changes, such IP's, ports etc...

For the development phase, the connection to EGIM equipment is virtually implemented through VPN connection, and this kind of communication is not as table as we would like. In a shortime, estimated by September 2016 the system will work in real conditions. From then, the whole configuration will be the same, and we will only need to change the network interface configuration on the Acquisition Server Virtual Machine.

We are trying to improve the system complementing the way that Data Management Platform should get the data. Initially, this configuration requires a connection polling to request data from DMP to SOS Server. This implies two operations for each request data. We have installed the Sensor Event Service in SOS server, with the objective that, users with a publish/subscribe-based interface, could access to sensor data and measurements located at SOS server. Basically, SES produces notifications and provides methods to subscribe for notifications and retrieve the latest notification. Meanwhile, users can also register new sensors dynamically and send notifications to the service.

This study benefited from H2020 INFRADEV--3--2015 EMSODEV Project n°676555 We acknowledge the support from CSIC-UPC Associates Unit Tecnoterra (Grupo de Desarrollo Tecnologico de Sistemas de Adquisición Remota Aplicado a Ciencias de la Tierra).

REFERENCES

[1] EMSO project site http://www.emso-eu.org/site/projects.html

[2] Arne Bröring, Christoph Stasch, Johannes Echterhoff "OGC® Sensor Observation Service Interface Standard", http://www.opengis.net/doc/IS/SOS/2.0 , 2012-04-20

[3] 52 North SOS 2.0 implementation, http://52north.org/communities/sensorweb/sos/

[4] Zabbix Monitoring System. http://www.zabbix.com/product.php

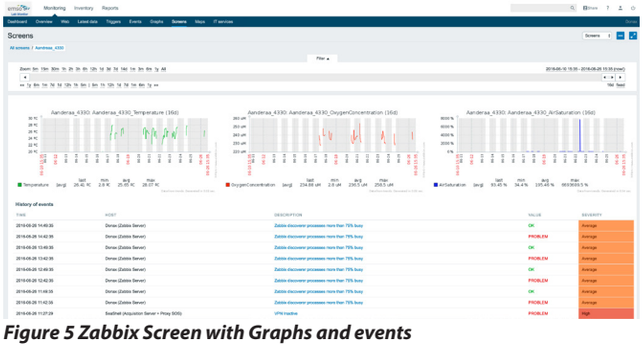

Figure 5 Zabbix Screen with Graphs and events